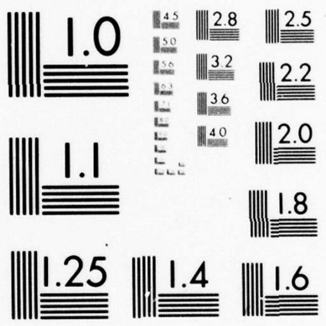
AD-A068 969 SPEECH COMMUNICATIONS RESEARCH LAB LOS ANGELES CA F/G 17/2
ACOUSTIC/LINGUISTIC ASPECTS OF AUTOMATIC SPEECH RECOGNITION.(U)
FEB 79 D J BROAD F44620-74-C-0034
UNCLASSIFIED AFOSR-TR-79-0525 NL

| OF |

AD
A068969

END
DATE
FILMED
7-79
DDC


1.0
4.5
5.0
5.6
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

FINAL SCIENTIFIC REPORT OF RESEARCH ON ACOUSTIC/LINGUISTIC ASPECTS OF AUTOMATIC SPEECH RECOGNITION

David J. Broad, Ph.D.

Speech Communications Research Laboratory, Inc.
806 West Adams Boulevard
Los Angeles, California 90007

Contract F44620-74-C-0034
from
Directorate of Mathematical and Information Sciences
Air Force Office of Scientific Research
Washington, D.C. 20332

Reporting Period: January 1, 1974 - December 31, 1978

February 23, 1979

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 18 AFOSR TR- 79-0525	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 ACOUSTIC/LINGUISTIC ASPECTS OF AUTOMATIC SPEECH RECOGNITION.		5. TYPE OF REPORT & PERIOD COVERED 9 FINAL rept.,
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 David J. Broad		8. CONTRACT OR GRANT NUMBER(s) 15 F44620-74-C-0034
9. PERFORMING ORGANIZATION NAME AND ADDRESS Speech Communications Research Laboratory, Inc. 806 West Adams Blvd Los Angeles, California 90007		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, Washington, DC 20332		12. REPORT DATE 11 23 February 1979
		13. NUMBER OF PAGES 49
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 16 2304 17 A2		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited. 12 51p.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

consonant
continuous speech
format frequency
fundamental frequency
interactive laboratory system

(continued on reverse)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

As word-level automatic speech recognition has improved, there has been an increased awareness of the need for more general procedures to recognize extended continuous speech utterances. This project has therefore addressed several problems related to the recognition of continuous speech. This has involved the development of a flexible interactive software system for acquiring, labeling, and analyzing continuous speech. This system has been used to complete most of the other studies reported here, including (1) a statistical procedure for the automatic extraction of reliable formant (cont'd)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE UNCLASSIFIED JOB

387 936

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Bolck 19 - continued -

inter-speaker normalization
piecewise-planar distributions
prosody
segmentation
spectral change
speech recognition
vowel
vowel duration

Block 20 - continued -

measures from large-scale data bases, (2) an algorithm for segmenting continuous speech on the basis of spectral change, (3) a study of vowel formant distributions in relation to the inter-speaker normalization problem, (4) various studies of the properties of vowels and consonants, and (5) studies of the prosodic patterns of speech, particularly as they can guide the recognition of extended utterances.

Contents

Page

This is the final scientific report on the project entitled Acoustic/Linguistic Aspects of Automatic Speech Recognition, which has been conducted under Contract F44620-74-C-0034 from the Directorate of Mathematical and Information Sciences of the Air Force Office of Scientific Research.

I. Introduction

The overall objective of research in automatic speech recognition is to carry on man-machine interaction in a mode of communication which is most natural to man, that is, speech. Automation has made such inroads into our way of life that not only is machinery used in many job functions but automated devices are frequently encountered in normal daily activities. Whether it be a telephone, cash register, computer, or an airplane, a person is usually required to supply manual inputs to the device in order for it to perform its functions. In many applications, man-machine interaction has reached such a level of complexity that it may be advantageous to have machines accept voice inputs.

During the flight of an aircraft there are times when the pilot is required to use eyes, hands and feet simultaneously in order to maintain proper control and guidance. Replacement of certain manual functions with voice commands could reduce human sensory and motor activities, which in turn reduces pilot fatique, yielding the combined effect of improved operational safety.

Thinking in terms of general applications involving the control or operation of machines, it can be seen that the machine side of the man-machine interface would become simpler if automatic speech recognition existed. There could be fewer knobs, switches, controls, etc. to cause confusion or hesitation in an emergency. The user would have more freedom of movement. It would not be necessary to be within arms length of the operator's console or remote switches, since instructions could be issued from anywhere in a room. Where continued visual concentration is required, commands could be given without having to look away at switches or controls.

Systems which normally require typed or keyboard inputs could become more efficient with voice input. The information transfer rate of speech is higher than typed input, therefore faster man-machine interaction would result.

As word-level recognition has improved, there has been an increased awareness of the need to recognize speech produced as continuous utterances - speech in which word boundaries are not easily detected in the speech signal and in which the pronunciations of the words themselves are altered by their running-speech context. Much of the work reported here has therefore been directed toward the study of continuous speech. This has involved the development of a flexible interactive software system for acquiring, labeling, and analyzing continuous speech data. This system, in turn, has enabled us to complete some of the other studies reported here. These include a statistical procedure for the automatic extraction of reliable formant-frequency measures from large-scale data bases, a study of the acoustical characteristics of reduced vowels, an algorithm for segmenting continuous speech on the basis of spectral change, a study of 2-dimensional constraints on the first 3 vowel formant frequencies and the relation of these constraints to the problem of inter-speaker normalization, and a study of how prosodic patterns, as realized by the contour of the fundamental frequency of the voice, can guide and assist the recognition and preliminary syntactic analysis of running speech. Part II of this report describes these results in more detail, while Part III lists the results so far published or submitted for publication. Part IV gives the names of all the personnel on the project.

II. Research Program

A. Interactive Laboratory System (ILS)

Early in the project considerable effort was directed toward implementing a set of mutually compatible computer programs for a variety of speech processing tasks. The set of programs is called the Interactive Laboratory System (ILS). It permits a user to record speech into a disk file, listen to any part of it, mark and label points and intervals of interest, and do various acoustic analyses, together with various statistical and graphics operations. The modular structure of the system is made possible by a set of conventions for standard formats of different kinds of files for waveforms, labels, and numerical data. These conventions are maintained by a common set of subroutines together with a standard allocation of common core storage for variables shared among programs, such as a starting frame number or the current sampling rate. This structure has permitted the development of a wide variety of programs which, when used in sequence, provide a truly flexible user-oriented means for analyzing speech. Thus instead of writing a new program for each new application, it is often possible to achieve the same result with a judicious sequence of ILS commands. And when a truly new application is required, then only the truly new part of the task usually need be written, because other parts, such as file descriptions, plotting, or statistical analysis, are already available as preceding or following commands. In fact, the system has proved to be so useful that it pays to write most new programs as ILS commands rather than as stand-alone programs in order to give the user the advantage of appending the other options to his application. By the same token, other ILS users then have access to an augmented system. The development of the basic ILS structure under support from the Air Force Office of Scientific Research has in this way benefitted other projects, and at the same time has given the present project more ready access to programs developed on other projects.

The development of speech label files has been a particularly useful part of the ILS development sponsored by the AFOSR. A speech label contains information on the phonetic category of any speech interval, together with its stress level, phonetic context, and the word in which it occurs. The label also specifies the location of the interval: its start frame, number of frames, and the name of the waveform file in which it is stored. Label files provide ILS with a sort of "indirect addressing" whereby a speech interval can be accessed by its name rather than by its location. It is also possible to do sorting on speech files by label specifications. Thus, e.g., one can automatically build up a file of all occurrences of some sound in a given context produced by some speaker.

Several laboratories around the world have implemented copies of the SCRL ILS system. This promises to enhance the exchange of ideas, programs, and data among researchers while simultaneously saving the cost of each laboratory rewriting the same applications programs in slightly different versions.

The details of the ILS system are given in "An Interactive Laboratory System for Research on Speech and Signal Processing" by Larry L. Pfeifer. (See Section III.)

B. Distribution Filtering

The collection of large scale speech data bases may require that the process of labeling relevant events in the acoustic signal be automated as much as possible. As a starting point for developing automated procedures we have investigated some characteristics of a manually labelled data base. This investigation was aimed at the following questions: 1) what type and magnitude of errors are present in acoustic measures on this data base, 2) what are the major sources of these errors, and 3) what can be done to reduce them.

The data base used for this study consists of 675 vowel tokens representing ten different vowel categories taken from a recorded interview with a single speaker. Vowels are labelled using a phonemic transcription. Transcription symbols were associated with the digitized acoustic waveform through an interactive process. The vowels were then analyzed for the frequency, bandwidth and amplitude of the first three formants by linear prediction analysis.

As a first step in characterizing the errors in acoustic measures on this data base, plots were made of the formant frequencies for all tokens of each vowel to observe trends in the data and gross excursions from these trends. While most data points had quite reasonable values for their respective vowels, there were several which departed radically from the central clusters. Possible sources of large errors which could explain the deviant samples are: 1) presence of inter-formant spectral peaks, 2) merged spectral resonances, 3) labeling errors, 4) transcription errors, or 5) improper location of the steady-state analysis window. Regardless of the source of error, inclusion of strongly deviant samples would obviously degrade pattern recognition performance. Therefore, they must be detected and either corrected or excluded from consideration. Even if there is no gross error resulting from any of the five main sources, there will still be some random error associated with the numerical estimate of a correctly identified spectral peak. This simple measurement noise, however, appears to be small in comparison to actual fluctuations in the speech signal, and is therefore not a major concern in the present context.

Variability may be characterized by the mean and standard deviation of formant frequency distributions for each vowel category. The possible sources of variability can best be explored by examining individual vowel tokens whose parameter values lie outside of some range. For this purpose, we adopted a technique called distributional filtering which rejects tokens having values greater than a certain number of standard deviations from the mean. A similar technique was used by Peterson and Barney[1] to detect measurement errors.

A basic question with regard to these filters is, what is the optimum number of standard deviations which will include most of the good measurements and exclude most of the wild samples resulting from the above five sources of error? To explore this problem, we tested six different ranges from 0.5σ to 3.0σ. Plots were made of the percentage of the original number of tokens which passed the filter as a function of percentage expected for normally distributed, independent parameters. Only the vowels /i/, /ɪ/, /ɛ/ and /ə/ were used for this portion of the sutdy because the greater number of tokens in these categories permitted more reliable statistics to be calculated for each filter condition. Despite differences in distributional characteristics, all four vowels showed similar trends, i.e., that filters with range less than about 2.0σ deviate markedly from expectation since far more tokens pass the filter than would be expected. This is probably attributable to the fact that the wild samples cause the standard

[1]Peterson, G. E. and Barney, H. L. (1952), Control methods used in a study of the vowels, The Journal of the Acoustical Society of America, Vol. 24, No. 2, pp. 175-184.

deviation of the original sample to be a poor estimate for that of the underlying populatin whith error points removed. We therefore chose 2σ as the range to study rejected tokens for sources of variability.

The 2σ filter rejected 75 out of a total of 501 tokens in the four vowel categories. When formant frequency measurements for the rejected tokens were examined in the context of surrounding sounds, it was found that 26 of these measurements could have been closer to population norms if the analysis window had been placed differently. This finding gives some indication of the need for more objective and perhaps automatic detection of the steady-state portions of vowels, since such an algorithm would have greatly reduced the variability of dispersion and skew in the data base under study. Twenty of these 75 probably had incorrect transcriptions. It was observed that 23 of the remaining 29 tokens were heavily coarticulated with nasals and liquids, thus indicating that most of the residue of large departures are not attributable to gross errors, but indicate real variability in the speech signal.

The work on distributional filtering has been presented in a paper "Labeling Speech Events for Acoustic and Linguistic Processing" by Robert J. Hanson and Larry L. Pfeifer.

C. Medium Continuous Speech Data Base

A data base consisting of approximately four minutes of natural speech from an extemporaneous interview with a male subject has been compiled for the purpose of studying speech sounds as they are realized in continuous speech. The speech was low pass filtered at 5 KHz and stored on disk for direct access. The overall digitized recording contains 2,016 phonemic units.

A description of the speech is provided by a transcription of the phonemic or quasi-phonetic values of each sound segment. A supplementary description and index of the data is provided by listings of 1) all words, in pronunciation order, 2) all words, with their corresponding phonemic transcription (which also itemizes alternate pronunciations), and 3) all phonemic elements, along with their phonemic environments.

The digitized speech files have been examined by means of the Interactive Laboratory System in order to determine the location of every word in the files. Words and word boundaries were located by combining listening along with time synchronized displays of the acoustic wave and corresponding formant trajectories. The word positions have been documented for further reference.

Our research efforts on this data base have concentrated on the vowel sounds. Manual segmentation has been performed to locate the vowel sounds within the data base, i.e., within the digitized files. The locations of all the stressed and reduced vowels, as well as the diphthongs, have been documented for further reference.

The occurrences of schwa have received particular attention. The steady-state (or quasi-steady-state) regions of these sounds have been analyzed to obtain formant frequencies amplitudes, and bandwidths. These data have been stored as records in organized files, where each record contains a label which identifies the schwa environments as well as notation on the presence of word boundaries. The details of this study were presented in the two papers "Some Acoustic Characteristics of Stressed and Unstressed Schwa" by Beatrice T. Oshika and Larry L. Pfeifer and "Some Acoustic Characteristics of Syllable Nuclei in Conversational Speech" by Michael A. Earle and Larry L. Pfeifer. (See Section III.)

The more general findings of this study have been published as "Acoustic Characteristics of Speech Sounds" by June E. Shoup and Larry L. Pfeifer. (See Section III).

D. Acoustic Discrimination of [f] and [θ]

In English, the distincition between the consonants [f] and [θ] (as in fought and thought) is difficult both perceptually and acoustically. It is therefore of interest to probe the limits of acoustic phonetic analysis by studying this distinction as a particularly challenging test case. Various algorithms were tested on a data base consisting of 48 occurrences each of [f] and [θ] as produced by one speaker. The signals were band limited to 5kHz. This may eliminate some phonetically useful information, but is nevertheless the most frequent bandwidth within which recognition systems are expected to function.

As would be expected, the discrimination for single frames was not as good as discrimination based on multiple frames within a segment. Discrimination using middle thirds of segments was superior to that using entire segments. This suggests that boundary and coarticulation effects degrade the discrimination by contributing variability to initial and final thirds of these consonants. Finally, it was found that a maximum-liklihood measure derived from the parameters of the statistical distributions of the spectral energy levels performed better than a simple spectral-distance measure. The best discrimination, about 89%, as achieved by applying a maximum-likelihood measure to middle thirds of segments was nearly identical to the score achieved by a phonetically trained human listener (90%). Given that the distributions of spectral energies for the two sounds do overlap, it may well be that this is close to the theoretical limit for discrimination of this sound pair, unless additional information from higher frequency bands or from transitions in adjacent sounds is also incorporated. This study is described in "Acoustic Discrimination Between [f] and [θ] in a Single Speaker" by David J. Broad. (See Section III).

E. Small Connected-Speech Data Base

1. Motivation. One of the main bottlenecks for developing effective procedures for the processing and recognition of continuous speech, especially unconstrained conversational speech, is the acquisition of carefully labeled speech data to test algorithms for phonetic analysis. A part of our effort has therefore been directed toward building up such a data base. This data base should be a valuable resource for the testing of any future algorithms for analyzing continuous speech.

To date, about 15 seconds of continuous speech has been analyzed according to the following procedures.

2. Marking Procedures. Using the Interactive Laboratory Systems (ILS) developed under support from AFOSR, a segment of speech of 1-3 minutes duration is stored on disk as a waveform bandlimited to 5kHz and sampled at 10kHz. The displayed waveform, formant frequencies, and rms energy are used together with repeated audio playback to assist the operator in making the best possible judgment for segmentation and transcription of each 100-frame interval (640 ms) of the waveform. Hard copies of the waveform and parameter display are preserved with their segmentation markers and phonetic transcriptions. At the same time, a label file is prepared which contains a greatly simplified form of the transcription. Each segment is marked as a V, S, N, C, or Z depending on its classification, respectively, as a vowel (V), sonorant constant (w, l, r, j) (S), nasal consonant (N), other consonant (C), or non-speech (silence or other non-speech) (Z). Ultimately, it would be desirable to encode the full phonetic transcription in the label file so that all the available phonetic information could be accessible to label-referenced algorithms. The labor involved for the present encoding system would make this not presently cost-effective.

3. Control on Subjectivity. Because even the best human transcriptions of connected speech contain some uncontrolled subjective factor, the above process is repeated on each speech segment by two transcribers working independently.

After a transcription is completed by both workers, disagreements between the transcriptions are noted and, by working together, the transcribers then resolve most of the disagreements by discussion and re-examination of the data. Usually, agreement is easily achieved. There is, however, always some residual disagreement or uncertainty in difficult parts of the transcription. These are left as points of ambiguity in the final transcription.

4. Consistency. A comparison of the two transcriptions for the same 15 second interval showed that one experimenter transcribed and labelled 125 segments, while the other experimenter transcribed and labelled 135 segments. While the number of segments differed by 10, the number of discrepancies between the two transcriptions was 19.

It is found that the two transcribers are fairly consistent with each other in their placement of segment boundaries. A one- or two- frame discrepancy is not uncommon, and the average placement is consistent to within about 10 ms. Specifically, 33 percent of the boundaries are in perfect agreement, 67 percent of the boundaries are within 6.4 msec or less, and 83 percent of the boundaries are within 12.8 msec or less.

These figures, then, provide a useful guideline for evaluating automatic segmentation algorithms: their agreement with human transcription need not be better than the agreement of the humans with each other. Note that the figure of about 10 ms is of the same order as a single pitch period of a male voice; it might therefore be taken to represent a measure of inherent uncertainty of event timing in speech.

F. Segmentation Algorithm

1. Description. In order to build up a substantial data base for meaningful studies of conversational speech data, it is expected that automatic algorithms will be necessary for segmenting and labeling speech events. One such algorithm has been devised for the automatic detection of segment boundaries. This is accomplished by computing the spectral variance of the speech signal as a function of time. The variance function tends to have local maxima in the transition region between sounds and local minima in sounds which can have steady-state characteristics. Thus a potential segment boundary is placed at the location of peaks in the variance function. All potential boundary markers are displayed on the graphics terminal for visual verification, but the operator must still identify and label the marked segments. This boundary detection algorithm is speaker-independent and operates reliably on unconstrained speech.

2. Human vs. Machine Marking. The performance of the automatic algorithm was evaluated at one level by comparing the location of vowel boundaries placed by the machine versus those placed by one of the transcribers. It was found that there was complete agreement on 33 percent of the initial vowel boundaries. Furthermore, 66 percent of the initial vowel boundaries and 52 percent of the final vowel boundaries were within 6.4 msec of each other. Also, 81 percent of the initial vowel boundaries were within 12.8 msec of each other.

Note that this is very close to the level of agreement between the two transcribers.

These results are very encouraging and demonstrate the effectiveness of the spectral variance function in locating vowel boundaries. It is expected that this algorithm could be the foundation for a totally automatic segmentation and labeling process.

G. Vowel-Formant Distributions and Inter-Speaker Scaling

In a previous study on another project, it was found that the first 3 formant frequencies for vowels produced by a single female speaker clustered around a 2-part piecewise-planar surface.[1] It was pointed out there that the question of how that result might generalize to other speakers was closely connected with the problem of the inter-speaker normalization of vowel formant frequencies. In the current project, therefore, the study was extended to five additional speakers to bring the total to 3 males and 3 females. In each case, a similar piecewise-planar representation was found for the first three formant frequencies, with separate planes being fit respectively to the front and back vowels. The orientations of these planes were similar from speaker to speaker, with the average displacement from the average orientation being only 10° for front-vowel planes and 13° for back-vowel planes. Though these angles are relatively small, the large

[1] D.J. Broad and H. Wakita, Piecewise-Planar Representation of Vowel Formant Frequencies, The Journal of the Acoustical Society of America, Vol. 62, No. 6, Dec., 1977, pp. 1467-1473.

sample size (about 5000 for each speaker) permits us to show that they are statistically significantly different from zero. Thus the different speakers have similar, but distinctly oriented representations. This fact is a sensitive test that allows us to reject the hypothesis of uniform formant-frequency scaling according to which the formant vectors for one speaker's vowels can be mapped onto those of another by simple scalar multiplication. A study of the positions of the planes (as distinct from their orientations) confirms this. It is interesting, however, that the speakers can be partitioned into two nearly-uniformly-scalable subsets, i.e., subsets within which departures from uniform scaling are still statistically detectable, but not substantial in a practical sense. This result tends to confirm Fant's notion of non-uniform scaling via partitioning the set of speakers into two uniformly scalable subsets (males and females.) More generally, the present results suggest that linear transformations on formant vectors (rotations, translations, and dot-product scalings) would be good candidates for inter-speaker scaling. This work is described in two papers by David J. Broad, "Piecewise-Planar Representation of Vowel Formant Frequencies Across Speakers" and "Piecewise-Planar Vowel-Formant Distributions and Inter-Speaker Scaling." (See Section III.)

H. <u>Prosodic Aids to Speech Recognition</u>

Prosodic aspects of speech (in English: stress, intonation, rhythm, and juncture) become especially important in connected discourse, especially for providing cues to the syntactic organization of utterances. Prosodic patterns in speech are realized through distinctive variations in the fundamental frequency of the voice (F_o contours) and in the durations and timing of events in the speech stream.

One of the problems connected with the analysis of prosodic information is that the prosodic parameters are functions not only of stress and intonation, but depend as well on the phonetic values of the segments and on their contexts. Thus, e.g., /i/ (as in bead) in English generally has greater duration than /ɪ/ (as in bid), and vowels preceding voiced consonants tend to be longer than those preceding voiceless ones (as /i/ is longer in bead than in beet). One of the results of the present project was a detailed study of these effects of context on vowel duration. It was found that the voicing of both the preceding and following consonants affected the vowel length, though the former effect was far the weaker. Similarly, effects were found for place and manner of articulation of the preceding and following consonants. Indeed, for the duration of a single vowel type, a simple additive-effects model seems to account for the context effects very well, while the inclusion of a simple scale factor suffices to track small variations in speaking rate. Errors of the model are comparable in magnitude to the purely random component of vowel duration. This work is detailed in a monograph by Ralph H. Fertig, "Temporal Interrelationships in Selected English CVC Syllable Nuclei." (See Section III.)

In previous research, Lea and his colleagues discovered a number of prosodic phenomena which can be useful for the recognition of connected speech. The distinctive high values of F_o and rms energy were used to construct an automatic detection for stress level, showing good agreement with listener judgments of stress. Detection of stressed syllables proves to be useful for the identification of "islands of phonetic reliability." It was also found that F_o contours and durations of silences could be used to locate syntactic boundaries in speech, thus assisting the initial analysis of an unknown utterance. Many of these interesting results have not been available in the literature, and a part of the current effort was therefore directed toward a reasonably complete and consistent review and description of prosodic aids for speech recognition. The findings are detailed in two papers by Wayne A. Lea, "Intonation and Segment Strings" and "Prosodic Aids to Speech Recognition." (See Section III.)

III. Publications and Presentations

The following publications and presentations with published abstracts have resulted from work on this project.

A. Publications

1. June E. Shoup and Larry L. Pfeifer, Acoustic Characteristics of Speech Sounds, in _Contemporary Issues in Experimental Phonetics_, edited by Norman J. Lass, New York: Academic Press, 1976, pp. 171-224.

2. Ralph H. Fertig, Temporal Interrelations in Selected English CVC Utterances, _SCRL Monograph No. 12_, Santa Barbara: Speech Communications Research Laboratory, 1976.

3. David J. Broad, Acoustic Discrimination Between [f] and [θ] in a Single Speaker, Conference Record 1976 IEEE ICASSP, New York: IEEE, 1976, pp. 162-165.

4. Wayne A. Lea, Prosodic Aids to Speech Recognition, in Trends in Speech Recognition, edited by Wayne A. Lea, Englewood Cliffs: Prentice-Hall, Chapter 8, (in press, 1979 expected), pp. 8-1 - 8-37.

5. Larry L. Pfeifer, An Interactive Laboratory System for Speech and Signal Processing (submitted for publication).

6. David J. Broad, Piecewise-Planar Vowel-Formant Distributions and Inter-Speaker Scaling (to be submitted).

B. Presentations with Published Abstracts

1. Beatrice T. Oshika and Larry L. Pfeifer, Some Acoustic Characteristics of Stressed and Unstressed Schwa, The Journal of the Acoustical Society of America, Volume 57, Supplement Number 1, Spring, 1975, p. S2. (Abstract)

2. Michael A. Earle and Larry L. Pfeifer, Some Acoustic Characteristics of Syllable Nuclei in Conversational Speech, The Journal of the Acoustical Society of America, Volume 58, Supplement Number 1, Fall, 1975, p. S96. (Abstract)

3. Robert J. Hanson and Larry L. Pfeifer, Labeling Speech Events for Acoustic and Linguistic Processing, The Journal of the Acoustical Society of America, Volume 60, Supplement Number 1, Fall, 1976, pp. S512-S513. (Abstract)

4. David J. Broad, Piecewise-Planar Representation of Vowel Formant Frequencies Across Speakers, The Journal of the Acoustical Society of America, Volume 64, Supplement Number 1, Fall, 1978, p. S180. (Abstract)

5. Wayne A. Lea, Intonation and Segment Strings, AAPS Newsletter, Volume 5, Number 2, December, 1978, (in press). (Abstract)

C. Previously Unreported

The following publication was based on work performed in a previous project (Contract F44620-69-C-0078) for the AFOSR, but was prepared too late to be included in that project's Final Scientific Report:

1. David J. Broad and June E. Shoup, Concepts for Acoustic Phonetic Recognition, in Speech Recognition, edited by D. Raj Reddy, New York: Academic Press, 1975, pp. 243-274.

IV. Personnel

The principal investigator on this project was David J. Broad. From 1974 through 1977, Larry L. Pfeifer served as co-principal investigator.

Wayne A. Lea, Beatrice Oshika, and June E. Shoup worked as Research Linguists on the project, while Steven B. Davis was a Research Engineer, and Robert Hanson was a Research Phonetician.

Jeffrey W. Andrews, Ted Applebaum, Donald P. Bollinger, Robert A. Dolan, Mark Krilanovoch, Jay Maskell, Gay E. Moore, Sharon Price, Paul E. Thurlow, and Robert Wohlford served as computer programmers on the project.

Timothy T. Burnett, Robert E. Hersey, Hajime Natsume and David L. Oster were computer operators and hardware technicians for the project.

Michael A. Earle, Ernest H. Hayden, Barbara F. Hanson, and Charlotte M. Wharton were research assistants. Special thanks are due to Sherry L. Francis, who volunteered her time as a research assistant during the winter and spring quarters of the 1978 academic year.

Judy A. Cherry, Jada A. Clark, V. Ann Forster, Martha Hinman, Mary A. Mines, Leeanna E. Reich, Margaret Retz, and Peggy J. Sanner served as technical typists on the project.

The present Final Report was typed by Betty J. Muiderman.

Appendix A. An Interactive Laboratory System for Research on Speech and Signal Processing.

AN INTERACTIVE LABORATORY SYSTEM FOR RESEACH IN SPEECH AND SIGNAL PROCESSING

Larry L. Pfeifer*

Speech Communications Research Laboratory, Inc.
800A Miramonte Drive
Santa Barbara, California 93109

June 1978

This work was supported by the Air Force Office of Scientific Research/AFSC, United States Air Force, under Contract F44620-74-C-0034. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

*The author is currently affiliated with Signal Technology, Inc., 15 West De La Guerra, Santa Barbara, California 93101.

INTRODUCTION

The use of digital computers in signal processing research has become a necessity, and software development is often a common problem. In laboratories supporting several different projects there can be duplication of effort due to incompatibilities among projects regarding data and program formats. Many laboratory computer systems have been programmed to operate in an interactive on-line mode but often they are done in assembly language, which has the drawbacks that it is more difficult to program, it takes more time to develop programs, and the software is machine dependent. This document describes the Interactive Laboratory System (ILS), which was designed to minimize these problems by using a high level language and establishing conventions in both program and data structures. The result is a highly modular system which can be easily programmed and in which both data and programs can be shared among projects. Furthermore, the system is effectively transportable so that future hardware or software upgrades do not seriously impede the computer support of research. The system also takes into account many aspects of efficient implementation and man-machine communication to provide effective support for research and development projects.

I. SYSTEM GUIDELINES

ILS as it exists today went through several phases of development. Throughout the evolution, however, there were underlying guidelines for the basic directions that were taken. These guidelines called for concepts which accounted for the needs of the computer system, the programmer, and the user.

The guidelines which were followed throughout ILS development are:

A. Operation of the system should be in an on-line interactive mode such that data can be easily input to (or output from) the system and readily accessed for verification, examination, and processing.

B. Program development should be in a high-level language in order to facilitate software implementation and promote machine independence.

C. The software should be modular in structure so that programs can be modified or inserted without affecting existing programs.

D. Programs should be invoked by means of logical procedures or commands, which minimize interaction time and which are user-oriented so that people can operate the system without first becoming computer experts.

E. Standards and conventions should be estabished so that programs and data can be shared, thus discouraging duplication of effort and simplifying the task of software maintenance.

A. Interactive Operation

The applications of ILS are focused on interactive processing. Experience has shown that interactive on-line communication has many advantages in a research environment because it offers the opportunity to make observations and select alternate courses of action in a more flexible manner than with batch processing. The Interactive Laboratory System is organized around a collection of inter-related command programs, each of which performs a specified function and can be executed by means of a simple keyboard initiation sequence.

B. High Level Language

An important feature in the design of the Interactive Laboratory System is that it was implemented in a high level language. Program development in assembly or machine language is more time consuming and results in system dependent software.

FORTRAN has been found to be a useful language for this purpose for several reasons:

1. Since some form of FORTRAN is available on most computers, the interactive laboratory system is, to a large degree, transportable from one computer to another, with only the insertion of a few system-dependent routines being required. This claim is supported by the fact that versions of ILS have been implemented on different models of the PDP-11 (under both the RSX-11 and RT-11 operating systems), as well as on Data General and IBM systems. A FORTRAN based system is helpful for importing programs as well as exporting them. Of course, FORTRAN compilers don't all follow the same standards so there can still be difficulties.

2. FORTRAN is a simple enough language that relatively complex programs can be implemented in a short period of time. Most scientific and research personnel know FORTRAN sufficiently well to write their own programs if necessary.

3. Algorithms can be tested and implemented in FORTRAN and later converted to assembly language versions if more speed and efficiency are necessary. This procedure has the further benefit of aiding transportability such that even if parts have been converted to assembler, equivalent FORTRAN versions are available. In laboratory systems, optimizing FORTRAN compilers such as FORTRAN IV PLUS on the PDP-11 and FORTRAN 5 on the ECLIPSE, have helped reduce the demand for assembly language programming.

Regardless of which language is used, implementation of an interactive system in a high level language has further long-term benefits. New releases of the computer operating system software are not likely to have a large impact on the interactive system. If there is a change from one operating system to another, within the same computer, the interactive system software can easily be transferred to operate efficiently within the constructs of the new operating system. The need to upgrade or change to a new computer system does not have a disasterous impact on the interactive system because it is basically a transportable package.

C. Modular Software

Overall system flexibility is achieved by means of modularity. Each command function stands on its own with the ability to take some form of input, possibly supplied by a previous command, and generate some form of output, possibly to be used by a following command. In order to obtain a particular result, in some cases, the user may have to execute a sequence of commands rather than one overall command. This initially appears inefficient, except for system considerations which make the approach very appealing. One strong point is that of reduced programming effort. If the outputs of three different commands can be used as inputs to any of three different subsequent commands, then nine possible combined operations are possible from only six programs. A second advantage is that of reduced program size. Each function program requires less memory and less disk space than equivalent functions which execute a command sequence as one program. If a particular sequence of commands is performed often enough, there may be sufficient justification for making an equivalent single command in order to reduce user interaction time.

In the situation where several projects are sharing the same computer, modular software allows project independence to be retained when necessary. This also relates to program development in general. In ILS, it is possible to develop, modify, and test a program without affecting the currently operational system and those using it. By defining some standards and conventions regarding file structures, different projects can share data and therefore further reduce duplication of effort. There are, of course, a certain number of programs which are project dependent but it turns out there is a large base of programs such as those for data collection, statistics, pattern recognition and graphics which can be shared by many if they follow a few conventions.

Holt [1] has pointed out that modularity in structure can also be necessary because of constraints such as limited memory size. This is the case in many laboratory systems, e.g., the PDP-11 computer is limited to 65 K bytes per user, in spite of the fact that much more memory than that can be attached to the machine. Even on large machines, if the ILS system was implemented as one program, with the commands as subroutines, the memory constraint would eventually be approached as the number of commands increased.

D. User-Oriented Operation

An important aspect in the design of ILS was to make the commands user-oriented so that operating the system does not require an engineering or computer background. This can be done with a standard terminal keyboard by using command names of one or more alphabetic characters which relate to the function which the command is to perform. A good combination of both brevity and clarity is necessary to avoid having to push extra buttons on the keyboard and yet prevent ambiguity. Further suggestions related to efficient man-machine interaction are given by Rouse [2] and Kennedy [3].

If the command function can be described in one word, the first letter of that word can be used as the command specification. When ambiguities arise, the first two letters of the word can be used, or perhaps the first letter from each word of a two-word command description can be used. For example, D for Display, L for Listen, or LR for List Records. There is no need to type the entire function word or words when an abbreviation will suffice. However, as the number of commands in the system increases, the amount of imaginative effort in making up unambiguous abbreviations also increases.

While many commands can be executed by typing everything on one line, this is not a requirement. The complexity of some programs is such that a user must be prompted with questions, or a menu of possible actions is displayed so the user can select a choice. The concept of being user-oriented is strongly related to user interaction time, therefore in order to reduce interaction time, questions or arguments are often made optional so that if no entry is made then a default entry is supplied by the system.

E. Standards and Conventions

The conventions which have been established for ILS are applied towards both programs and file structures. The conventions help to insure compatibility between programs and data, and this is the key to the concept of a shared system. Four of the most important conventions are discussed in the remainder of this section.

1. Documentation

ILS is a user-oriented system, and one of the things that makes it so usable is documentation. The new user benefits from having manuals which tell what the commands do, how they do it, and what buttons to push to get action. Having each user be self-sufficient is of great importance because it reduces repeated explanations by those who write the programs. As the ILS system grows and the number of commands increases, documentation becomes important to the experienced user also, e.g., for trying some new command or recalling an old one which hasn't been used for a while.

One of the more important documentary items is the ILS Users Guide. This is a 450 page manual describing the various ILS commands, along with examples and illustrations. Conventions have been established in the format of command documentation so that information of the same type can always be found in a consistent place in the documentation.

An installation guide is included with each distributed ILS system. This guide describes the distribution files, the installation procedure, the trial installation of an ILS subset, and a full ILS installation.

In addition to the written documentation there is an abbreviated form of on-line documentation which is provided by a HELP command in ILS. This form of documentation makes it possible for people to use the system without always having a users guide by their side.

2. Variable Names

A set of names has been adopted for variables which are frequently used throughout the system, and the use of these names is encouraged so that there may be some consistency among programs. This serves the purpose of aiding the programmers in software development and maintenance. A person familiar with the conventional names has an easier time reading programs. Because the command programs of ILS can be developed independently, the names convention is difficult to enforce without having a reviewer, and this type of software support is not available at many research laboratories.

3. Subroutines

Existing subroutines are used as much as possible throughout the system. Many often-used functions, for example, data transfer to and from disk, have been put in subroutine form and programmed for efficient operation. Subroutines which are used by more than one command program are placed in an ILS library. All programs and subroutines are kept in one common place so that anyone can examine or copy them. However, they should be protected so that only designated people can update or delete them. Since the more routine ILS tasks already exist in subroutine form, programming effort can concentrate on the application rather than on ILS overhead.

4. File Format

The one convention which has been most instrumental in promoting program and data interchange is the use of standardized file structures for storing data. Four types of files have been defined in order to meet the needs of different signal processing applications. The four types are: 1) sampled data files, 2) analysis files, 3) record files, and 4) label files. The overall format of the first three types of files is shown in Fig. 1. These are basically data files, each with a 64 word file header containing information which is global to the file. Conventions have been set down on what gets stored in the header and where it's located. Element 63 of the header, for example, is reserved for a code which identifies the type of file. This provides a means of preventing accidental destruction of valuable data. For example, sampled data is considered to be expensive, and intended primarily for reading only, therefore, very few programs are permitted to write into a sampled data file unless it has been explicitely "unprotected" by the user. There is no restriction on file length other than it must fit on the disk. Since the files are accessed in a random fashion they must reside on a random access file structured device. There are standard ILS subroutines available for accessing these files in an efficient (buffered) manner, so the ILS programmer is not burdened with the details of file input and output. The individual format of each file will be discussed separately.

Sampled Data Files

As the name implies, these files are used for storing sampled data. The data is stored in integer format and could come from such sources as the analog-to-digital

COMPUTER WORD NUMBER	SAMPLED DATA FILE	ANALYSIS FILE	RECORD FILE
1	no. points per analysis window	no. points per analysis window	next record number
2	no. analysis coefficients	no. analysis coefficients	number words left in file
61	file header len.	file header len.	
62	sampling freq.	sampling freq.	
63	-32000	-29000	-30000
64	init. flag	init. flag	0
		analysis vector 1	directory pointers for direct access
	sampled data	analysis vector 2	record 1
			record 2
	(integer)	(128 16-bit integer words per vector)	(floating point, integer, or ASCII data)
	variable length file	variable length file	variable length file
		analysis vector N	record N

Figure 1. Basic format of three ILS file structures.

converter (single or multi-channel), from some computational or simulation process such as a speech synthesis program, or from another sampled data file, such as when transferring segments from one file to another (digital splicing). When the data comes from the analog-to-digital converter, the samples are stored sequentially, one per word, until the file is full. The amount of speech which can be digitized is a function of the length of the file and the sampling frequency. In our system, in order to meet the demands of sampling frequencies above 20 kHz, these files must be contiguous (composed of consecutively numbered disk blocks) if they are to be used in an analog-to-digital or digital-to-analog process. Otherwise they need not be contiguous for any other operation within ILS. Sampled data files can be created at will by the user, as long as there is storage space.

Analysis Files

This type of file is used to store integer data from different analysis programs. Analysis outputs are stored as vectors of length 128, with one vector for each consecutive window of analysis. For speech processing applications, conventions have been established for the position of data within a vector, such as linear prediction coefficients, formant parameters, RMS energy and fundamental frequency. Analysis files are of variable length depending on the amount of data analyzed. If an analysis file happens to be too short when an analysis is started then the file is automatically lengthened. Thus, the user does not have to be concerned about keeping track of the length of analysis files.

Record Files

Record files were incorporated into ILS to provide a more flexible data structure than was provided by sampled data or analysis files. A record file contains a set of records and a directory. The directory contains the storage position of each record in a file so that rapid sequential or random access is possible. Records can store data in integer, floating point, or ASCII format. Each record has a header as well as its body of data, both of which are variable in length. The record header provides a means of tagging the record data with descriptive information.

There are additional levels of data structuring within a record which results in efficient storage and data organization. Each record can contain a variable number of items and each item contains a specified number of elements.

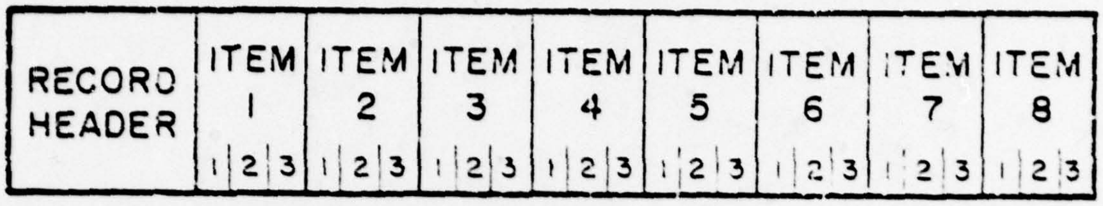
Various combinations of item and element structures are shown in Fig. 2. The four records shown are all the same length but have different data structuring. There is no restriction on the combinations other than there must be at least one item per record and at least one element per item. Since a record header represents a certain amount of storage overhead, the item structure effectively allows the packing of several records under one header and thereby contributes to efficiency. ILS commands can access data at the record, item, or element level, so there is complete flexibility in data storage.

Because of their flexibility, record files have become the foundation on which many commands operate for information storage and retrieval. Commands exist for performing printing, plotting, statistics, and pattern recognition using data in record files. Likewise there are many commands which store their results in record files. Thus, a record file is like a node which has many input branches (which write records) and many output branches (which read records).

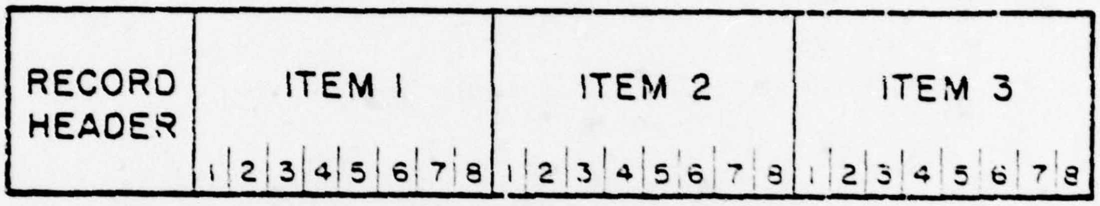
Label Files

A label file is an all ASCII file which is used to document events occurring in a sampled data file.

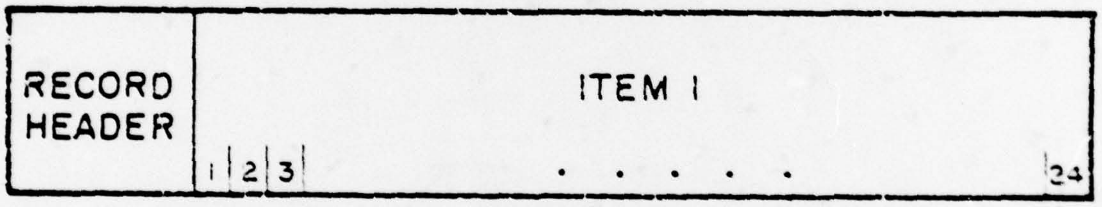
Besides documentation, labels provide a means for automatic retrieval of labeled events. Programs can read each label, locate each event, and perform specified operations e.g., feature extraction for pattern recognition experiments. Sort keys can also be specified so that only selected labels (or their corresponding events) are retrieved. A label consists of two consecutive lines of text with up to 72 characters per line. Each label contains descriptive information about an event and the location (position) of the event in the sampled data file. Any size or kind of event can be labeled. A label file can be printed or edited, without special software, on any computer which has internal ASCII coding. There is no restriction on what can actually be labeled, or how the descriptor fields are used, or what's put in them (except two of the fields are expected to contain numeric characters). Projects can use the labels to suit their own needs. Any number of the fields can be omitted if not needed. Because each label is a complete descriptor of an event, the events can be labeled in any random order and they can be in different sampled data files.



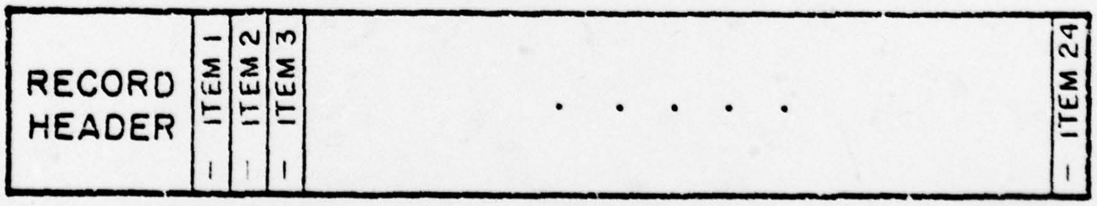
A. 8 ITEMS, 3 ELEMENTS/ITEM



B. 3 ITEMS, 8 ELEMENTS/ITEM



C. 1 ITEM, 24 ELEMENTS/ITEM



D. 24 ITEMS, 1 ELEMENT/ITEM

Figure 2. Examples of different data structures allowed within the records stored in record files.

II. OPERATIONAL CONCEPTS

A. Global Accumulator

A traditional concept which is often implemented in similar systems of this type is the use of an accumulator into which data is loaded and then some sort of processing performed, with the results possibly being stored back into some file. In some signal processing applications, sampled data files can be very large and the process of loading a file into an accumulator would be time consuming and would require many data transfers. Likewise, the size of a file might be constrained by the size of the accumulator, where an in-core accumulator might be considerably smaller than a disk-resident accumulator.

The ILS system makes use of what could be referred to as a global accumulator. This means that rather than loading a file into an accumulator, a file simply becomes the accumulator, and the specified file is referred to as the primary file. A second file can also be specified to the system so that results of computations on the primary file can be stored in the secondary file. The general information transfer path is to input data from the primary file and to output to the secondary file. Being a primary or secondary file is not a permanent attribute of a file, but rather a designation which is retained by the system until a new primary or secondary file is specified. Thus any file can be the primary or the secondary, depending upon how it is specified by the user. Files are processed in-place on the disk with computation being done in memory on small portions at a time.

B. Variable Frames

The ILS system relies largly on the access of data stored in disk files. The sampled data files are made up of sequentially stored data so a means of specifying data locations was devised. Disk files are typically made up of a series of disk blocks which are numbered sequentially with respect to the beginning of the file, and each block may contain, for example, 256 words. It is often desirable to access only certain portions (segments) of the sampled data, but the desired data may not start on a disk block boundary or be contained within some integer number of disk blocks. Therefore, software has been implemented to provide for data access in terms of variable length units referred to as frames. The length of a frame can be defined by the user according to what is convenient for the application at hand. If the frame size is set to one, then references to the data are made in terms of single sample points. If the data had been digitized at a sampling frequency of 10,000 Hz then a frame size of 100 would correspond to 10 msec and the user could refer to the data in time units of 10 msec. The length of a frame is stored internally in ILS so that all command programs have access to the current frame size.

Since all files in the ILS system have file headers, frame one of the data begins immediately after the header. The location of data in a sampled data file is usually specified in terms of its starting frame and number of frames. This is a convention which is followed throughout ILS and many commands require the starting frame and number of frames as data specifiers.

C. Command Format

The basic ILS command format is modeled after that described by Greaves [4], and it consists of three parts:

1. Command specification
2. Alphabetic arguments
3. Numeric arguments.

For convenience, this information is often typed on one command line terminated by a <RETURN>.

The command specification is a one or two letter command name, which is the name of the desired program. The single and double letter combinations avoid conflicts with operating system commands and system utilities which usually have names of three or more characters (in the PDP-11 RSX-11D operating system). A <SPACE> is used to separate the command specification from the alphabetic and numeric arguments.

Alphabetic arguments are useful for specifying options in a command program. Thus, rather than having several separate command programs which perform minor variations of a particular task, a single command program can have alternate execution steps depending on the arguments. For example, the D command is the basic command for displaying sampled data in the primary file. But by typing D S, the data will instead be displayed from the secondary file, or if D E is typed, the screen will be erased prior to the display of the data. Thus, the single program D provides a variety of services and therefore helps to minimize the proliferation of command programs.

Alphabetic arguments are specified prior to numeric arguments. They can be concatenated on the same command line but are not interspersed between numeric arguments. There is no delimiter between the alphabetic arguments so there must be no ambiguities which cannot be resolved when arguments are concatenated.

The purpose of numeric arguments is to provide the user with a means of supplying variables to a command program. This contributes toward making programs general purpose so that they can have broader applications. Numeric arguments are typed on the command line after the alphabetic arguments (if there are any). The number of numeric arguments is a function of the needs of the command program and the predefined length of the argument list. There is, however, some practical limit on the number of arguments a user can remember without continually referring to documentation.

The command programs expect the numeric arguments to be in a particular sequence so the user must enter them in the proper order. If there are any restrictions on the numeric values which can be accepted, it is the responsibility of the program to check for illegal or undefined values. As a result, the user does not have to be so careful, and if some mistake is made on entry, the program can exit gracefully with a message to the user, rather than attempting to perform erroneous or meaningless computations.

The arguments in a command line are passed on to the command program itself. The command program may not need any arguments, thereby ignoring any additional typing beyond the command specification. If the command program does require arguments, then it can examine the argument list to obtain them. If a required argument is missing from the list (presumably intentionally) then some default argument value should be supplied by the program itself. The program should also test for illegal arguments.

Arguments are placed in an argument list within ILS so the command program can access whatever it needs. Upon completion of the program, the arguments are moved to a second internal argument list so that the next command can access the arguments of the previous command if necessary. Whenever possible, all the arguments are provided on a single command line so that once a command program is started it can proceed without interruption.

III. SYSTEM COMPONENTS

The ILS system is organized around three functional components consisting of:

1. Command interpreter
2. COMMON storage
3. Command program.

Each component can be implemented in different ways depending upon the size of the computer, its resources, and the operating system. A description of these components and how they are currently implemented in a PDP-11 running RSX-11D will be presented in this section.

A. Command Interpreter

An ILS command is invoked by typing the necessary information on a single command-line. The processing of a command-line is done by a command interpreter program. One of the functions of the interpreter is to extract the command name. The command name is actually the name of a program which performs the desired function. The command interpreter must then check to see if the specified command program exists. If the program exists, it is loaded and the command is executed. If the program does not exist, then a message to that effect is printed on the user terminal, thus indicating that no such command has been defined or installed in ILS.

Operating systems which support on-line terminals already have a command-line interpreter for processing inputs from a user. There are, unfortunately, many system-supplied command interpreters which do not operate in the manner described above or which do not accept the ILS command-line format. When such is the case, then the interactive system must have its own resident command-line interpreter. This does not have to be a large program but it means that an ILS user has a program running at all times.

There are at least two operating systems which do have command-line interpreters which are compatible with the ILS format. One is RSX-11 for the Digital Equipment Corporation PDP-11 and the other is RDOS for the Data General Eclipse. In either operating system the interpreter attempts to load the program whose name has been typed. If the name is followed by a space, then additional information can be entered on the same command-line. In the RSX-11 operating system, the entire command-line is saved in a memory buffer and is accessable to a user program. In the RDOS operating system, the command-line is saved in a disk file, which can also be accessed by a user program. Thus, with the proper system-supplied interpreter, there are no ILS programs in memory except when an actual command is being executed.

B. Common

The information stored in COMMON is required at all levels of operation of the ILS system. Definitions or constants related to the computer system and ILS are stored for access by all programs. COMMON is used to retain the status of ILS, such as the names of the current primary and secondary files. There is also reserved and unreserved array space in COMMON so that data can be conveniently passed from one command to another.

The conventional application of COMMON is as a memory resident storage region which can be accessed by independent programs or subroutines which may or may not be overlayed or chained. ILS follows this convention while programs are running but goes one step farther by also keeping a copy of COMMON in a disk file, thereby releasing valuable memory resources when programs are not in execution. This adds one or two extra disk operations to every ILS command, but in a memory bound computer system disk transfers have a smaller impact on system performance than memory utilization. The number of extra disk transfers varies between one and two depending upon the command program. All programs must read the COMMON file, but if the program changes nothing in COMMON there is no need to write it back out to disk. Since COMMON is stored in a disk file, there is an advantage that the user can leave the computer and return at any time later to continue exactly where he left off.

Each user has his own COMMON file which is stored under his own directory or user area. There is only one copy of the ILS system but any program can be in execution by more than one user if ILS is run on a multi-user system. Each program first determines the directory of the current user and then loads the COMMON file from that user's directory. The COMMON region of storage is a contiguous block of memory, therefore the entire block can be transferred in one operation. The single variables and reserved arrays are declared first in COMMON and the unreserved arrays are declared last. As a result, in order to improve efficiency, if the unreserved arrays are not declared and used in the command program, they are not included in the transfer of COMMON to or from the disk.

C. Command Programs

Each ILS command is actually a program which has the same name as the command. The command programs are kept resident on disk and are loaded on demand. Each command program typically has COMMON declarations which conform to ILS conventions. The first thing a command program does is read COMMON from the disk in order to get the command arguments and anything else it may need. Upon completion of its task a command program typically writes COMMON back to disk. There are no restrictions or limitations imposed on command programs in terms of system resources or implementation techniques. There are many standard ILS subroutines available to perform often used operations. Basically, any program which is allowable within the computer system is allowed within the ILS system.

A summary of the ILS commands is given in Appendix A. They are listed alphabetically by the one or two character command name and a one-line description. Space does not permit a detailed description of each command (the ILS users guide is approximately 450 pages) but the documentation of the EF (Elliptic Filter) command is given in Appendix B.

The current set of ILS commands performs a variety of functions related to signal processing, statistics, pattern recognition, graphics, and speech analysis. Many commands act as building blocks for other commands such that a complex process can be performed by the proper sequence of commands.

REFERENCES

1. R.C. Holt, "Structure of computer programs: a survey", Proceedings of the IEEE, vol. 63, pp. 879-893, June 1975.

2. W.B. Rouse, "Design of man-computer interfaces for on-line interactive systems", Proceedings of the IEEE, vol. 63, pp. 847-856, June 1975.

3. T.C.S. Kennedy, "The design of interactive procedures for man-machine communication", Int. J. Man-Machine Studies, vol. 6, pp. 309-334, 1974.

4. J.O.B. Greaves, "The bugsystem: the software structure for the reduction of quantized video data of moving organisms", Proceedings of the IEEE, vol. 63, pp. 1415-1425, October 1975.

5. J.D. Markel and A.H. Gray, Jr., Linear Prediction of Speech. New York, New York: Springer-Verlag, 1976.

6. A.H. Gray, Jr. and J.D. Markel, "A computer program for designing digital elliptic filters", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-24, pp. 529-538, December 1976.

7. J.H. McClellan, T.W. Parks and L.R. Rabiner, "A computer program for designing optimum FIR linear phase digital filters", IEEE Trans. Audio Electroacoustics, vol. AU-21, pp. 506-526, December 1973.

8. H. Williamson, "Hidden-line plotting program", Comm. ACM, vol. 15, pp. 100-103, February 1972.

9. A.H. Gray, Jr. and J.D. Markel, "Distance Measures for Speech Processing", IEEE-ASSP, vol. ASSP-24, October 1976.

APPENDIX A
Summary of ILS Commands

A	Inverse filter analysis (autocorrelation method) [5]
AC	Inverse filter analysis (covariance method) [5]
AP	Inverse filter analysis with pitch extraction
AQ	Analysis coefficient quantizing test
AS	Assign logical unit numbers for program input/output
B	Best fit pattern recognition
C	Cursor
CL	Cursor label for pitch synchronous analysis
CO	Hard copy
CP	Cepstrum display
CS	Compute statistics on sampled data file
CT	Context (data points/frame)
D	Display frames
DF	Distance finder
DG	Display parameter with grid
DI	Distance measure from frame to frame [9]
DP	Display parameter
EA	Equal area quantizing test
EF	Elliptical filter design [6]
ER	Erase the screen
F	Frequency plot of smooth spectra
FD	Frequency spectrum display
FI	General purpose digital filter
FT	Formant tracker
FU	Formant tracker with smoothing
GR	Grid for spectral plots, or axis for Cepstrum display
HE	File header length
HH	Help with ILS commands
HI	Histogram plot of record data
I	Initialize file for analysis
ID	Initialize or update user buffer parameters
ILS	Create and initialize user buffer
KK	Spectral peak processing
L	Listen to sampled data through D/A converter
LB	Label a segment
LF	FIR linear phase filter design [7]
LL	List Label file
LR	List the contents of record files
M	Move frames
MO	Modify sampled data
MP	Modify analysis parameters in record data file
MR	Move records
NS	Add noise to sampled data file
O	Open files for records
P	Print frames
PA	Pitch synchronous analysis of speech data
PC	Compute principal components from record files
PF	Filter records which deviate from a mean record
PL	Plot record file data on terminal screen

PN	Synthesize speech from pitch sync. analysis params.
Q	Feature extraction from labeled sampled data segments
R	Record data through A/D converter
RA	Residue analysis on output of A command
RC	Residue analysis on output of AC command
RL	Copy parameters from label file into user buffer
RS	Root solving formant finder and normalization
RV	Reverse the order of sampled data points
S	Set and test sampling frequency
SA	Spectral resonance analysis on output of A command
SC	Set the value of the sector number in the user buffer
SE	Sort records by environment code
SI	Analysis with pitch extraction using SIFT algorithm
SL	Sort labels into time-increasing order
SM	Calculate statistical measures on record data
SN	Synthesize speech from reflection coefficients
SP	3-dimensional spectral plotter [8]
SR	Running statistics on record files
ST	Saturation test on sampled data after A/D conversion
T	Transfer frames from one file to another
TD	Time and date displayed on terminal screen
TF	Test function - set up numerator/denominator terms - synthesize specified signal
TL	Select specified labels
TM	Tic mark
TT	Transfer sampled data with or without labeling
UI	User identification code (UIC)
UP	Unprotect a sampled data file
V	Verify analysis conditions
VA	Variable window analysis using covariance method
VD	Variable distance threshold evaluation
VT	Plot a vocal tract cross section on display screen
W	Select, create, or delete WD files
WL	Select a label file
WR	Write records into file from keyboard
X	Inverse filter spectrum
XP	Expand sampled data file
XT	Find min. and max. values of data in a record file

APPENDIX B

Example of EF Command

```
                                          ***********
ELLIPTIC FILTER COMMAND                   *EF       *
                                          ***********
```

Function

EF Elliptic Filter Design

Command Format

```
EF n1,n2,n3,n4,n5,n6<RETURN>
```

Alphabetic Arguments

None

Numeric Arguments

n1=	Order of filter (in S-plane)
n2=	Pass band ripple (in milli-dB)
n3=	Sampling frequency (in Hertz)
n4=	Lower pass band limit (in Hertz)
n5=	Upper pass band limit (in Hertz)
n6=	>0, stop band limit (in Hertz) <0, stop band attenuation, "dB-down" (in dB)

Preparatory Commands

None

Confirmatory Commands

TF Command. By means of TF 4, plot the log spectrum of the discrete transfer function, P(z)/A(z).

Description

The EF command designs an elliptical filter as close to the input specifications as possible. The numerator and denominator coefficients are stored in COMMON. The design is printed out at the end of the program (refer to Figure EF-1). It may be plotted out with the TF 4 command (refer to Figure EF-2) and then used with such commands as AP, FI, etc.

The EF command can design any of the following four types of filters depending upon n4, n5 and n6:

Low Pass Filter:

Enter $n4 < 1 < n5 < n3/2$ with n6 as the stop band limit.

High Pass Filter:

Enter $0 < n4 < n3/2 < n5 + 1$ with n6 as the stop band limit.

Band Pass Filter:

Enter $0 < n4 < n5 < n3/2$ with n6 in the stop band.

Band Reject Filter:

Enter $0 < n5 < n4 < n3/2$ with n6 in the stop band.

Note: n6 may be entered as the stop band attenuation or the stop band limit for any of the four cases.

Example

```
MCR>EF 4,200,10000,0,800,-30
```

This is an example of a fourth order low pass filter ($0 < 1 < 800 < 10000/2$). The ripple is .2 dB and the stop band attenuation is 30 dB. The limits of pass band are 0 Hertz and 800 Hertz. The sampling frequency is 10000 Hertz. Figure EF-1 shows the output of EF command including the numerator and denominator coefficients. Figure EF-2 shows a picture of the filter over the frequency spectrum using the TF 4 command (refer to the TF command).

(February, 1978)

```
EF      4   200 10000      0    800    -30

 LOW PASS FILTER          0.200 DB RIPPLE
 PASS BAND LIMIT        800.000 HZ
 STOP BAND LIMIT       1123.813 HZ
SAMPLE FREQUENCY      10000.000 HZ
THETA=        44.168324, RHO=          0.212150,         -29.999996 DB DOWN
  4   = ORDER
         A(J)                 P(J)
 1    0.100000000E+01    0.399585851E-01
 2   -0.301406553E+01   -0.723931193E-01
 3    0.367655301E+01    0.999343097E-01
 4   -0.210091043E+01   -0.723931193E-01
 5    0.474324971E+00    0.399585813E-01
POLES
       0.801583         0.475278
       0.705460         0.220266
ZEROS
       0.735601         0.677415
       6.170251         0.985401
  2 -0.907083E+00   0.607617E+02 -0.301409E+01   0.907089E+00
  3  0.956559E+00   0.516446E+01  0.367655E+01   0.956559E+00
  4 -0.866117E+00   0.129030E+01 -0.210091E+01   0.866117E+00
  5  0.474325E+00   0 100000E+01  0.474325E+00   0.474325E+00
MCR>
```

Figure EF-1. Example of printout obtained after the design of a filter with the EF Command.

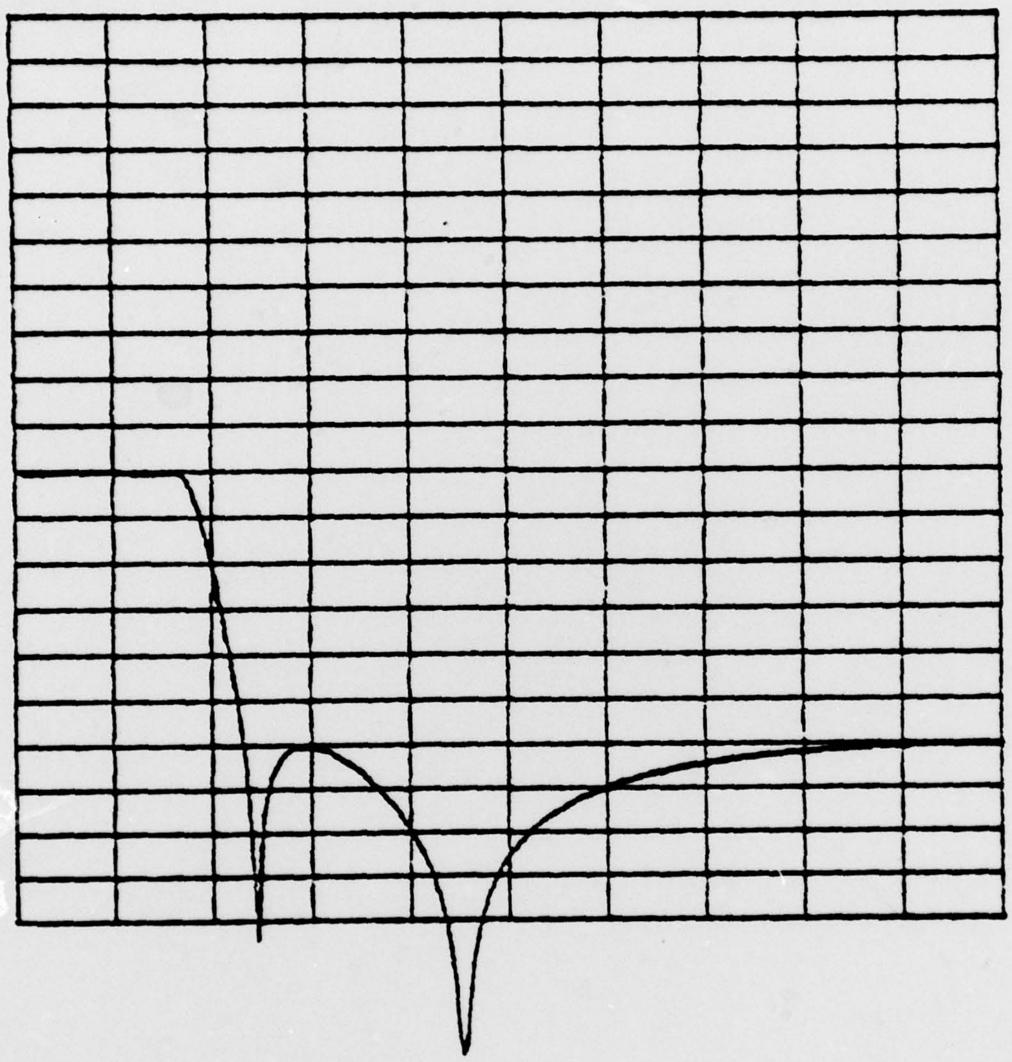


Figure EF-2. Frequency response of filter designed with the EF Command.